APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/676,889	09/30/2003	Shih-Wei Liao	42P16806	8089

8791 7590 02/20/2009
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
MITCHELL, JASON D	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
02/20/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/676,889	LIAO ET AL.	
	Examiner	Art Unit	
	Jason Mitchell	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 December 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-29* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-29* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *08 December 2008* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This action is in response to a request for continued examination filed on 12/8/08.

Claims 1-29 are pending in this application.

Response to Arguments

Applicant's arguments filed 12/8/08 have been fully considered but they are not persuasive.

In the 1st full par. on pg. 13, the applicants state:

> Rather, Luk provides software-controlled pre-execution schemes to help programmers writing memory-friendly programs using pre-execution threads and for compiler to insert pre- execution (Luk, page 40, col. 2, par. 4 - page 41, col. 1, par. 1). Luk proposes ISA (Instruction Set Architecture) extension with three new instructions to spawn a pre-execution thread at a particular PC (Program Counter) and stops when certain number of instructions have been executed in the pre-execution (Luk, page 44, col. 1, pars. 4-5, col. 1, par. 1). According to Luk, a compiler inserts pre-execution by determining which references could benefit from the pre-execution and calculating the pre-execution distance (Luk, page 44, col. 2, par. 4). In addition, Luk discusses performing locality analysis to locate where cache misses occur in a program (Luk, page 51, col. 2, par. Phase 1, Step 1). Specifically, Luk demonstrates code fragment tmp->next for determining data access for locality analysis (Luk, page 51, col. 2, par. Phase 1, Step 3). Thus, Luk's analysis identify which data access instruction in a program would benefit from a thread pre-execution. However, Luk is completely silent about selecting a code region from one or more code regions sharing instructions in source codes of a main thread.

The examiner respectfully disagrees. In addition to identifying the data access instruction would benefit form a thread pre-execution (e.g. cause a cache miss) Luk discloses identifying the source code which would generate the necessary address (e.g. pg. 44, col. 2, 3rd full par. “calculates the pre-execution distance”; Appendix, Phase 1, Step 2 “control-flow analysis and call-graph analysis [24] can easily discover multiple paths and procedure calls"). Further, the claims do not specify what constitutes one of the "code regions", specifically the claims do not indicate how or why a particular series of instructions is considered a region.. Accordingly Luk meets the broadly claimed limitation.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1-2, 8-9, 15-16 are rejected under 35 U.S.C. 102(b) as being anticipated by "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk).

Regarding Claim 1: Luk discloses a method, comprising:

analyzing source codes of a main thread having one or more delinquent loads, the one or more delinquent loads representing loads which likely suffer cache misses during an execution of the main thread (pg. 44, col. 2 3rd full par. "locality analysis phase which determines which references are likely to cause cache misses"; also see Appendix Phase I, Step 1; pg. 45, "data address generation and the surrounding control flow"), the source codes including one or more code regions, each code region corresponding to a sequence of instructions in the source codes, the one or more code regions sharing at least one instruction in the source codes (e.g. Fig. 1(d); note that without indicating what constitutes a 'code region' the claim is so broad that any source code necessarily contains appropriate regions);

selecting a code region from the one or more code regions for one or more helper threads with respect to the main thread based on the analysis (pg. 44, col. 2 3rd full par. "locality analysis phase which determines which references … could benefit from pre-execution"; also see the Appendix Phase I, Step 2 "control-flow and call-graph analysis"); and

generating code for the one or more helper threads, the one or more helper threads being speculatively executed in parallel with the main thread to perform one or more tasks for the region of the main thread (pg. 44, col. 2, 3rd full par. "performs all necessary code transformations"; also see the Appendix Phase II).

Regarding Claim 2: The rejection of claim 1 is incorporated; further Luk discloses identifying the region comprises:

generating one or more profiles for cache misses of the region (pg. 43, the par. bridging the cols. "based on profiling information"; also see the Appendix, Phase I, Step 1 "This step can be accomplished through some low-overhead profiling tools"); and

analyzing the one or more profiles to identify one or more candidates for thread-based prefetch operations (pg. 43, the par. bridging the cols. "the compiler usually needs to heuristically decide how to prefetch … based on profiling information").

Regarding Claims 8-9: Claims 8-9 recite a computer readable storage medium for instructing a computer to perform the methods of claims 1-2 and are addressed similarly.

Regarding Claims 15-16: Claims 15-16 recite a system for performing the method of claim 1 and are addressed similarly.

Additionally claim 16 recites and Luk discloses the process is executed by a compiler during a compilation of an application (pg. 44, col. 2, 3rd full par. "the compiler … is responsible for inserting pre-execution").

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 3-4 and 10-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk) in view of "Exploiting Hardware Performance Counters with Flow and Context Sensitive Profiling" by Ammons et al. (Ammons).

Regarding Claim 3: The rejection of claim 2 is incorporated; further Luk discloses generating one or more profiles for an application (pg. 43, the par. bridging the cols. "decide how to prefetch … based on profiling information") but does not explicitly disclose executing the application with debug information or sampling cache misses and accumulating hardware counters for each static load.

Ammons teaches generating one or more profiles comprises:

executing an application associated with the main thread with debug information (pg. 86, col. 1, last full par. "a tool … that instruments program executables"); and

sampling cache misses and accumulating hardware counter for each static load of the code region to generate the one or more profiles for each cache hierarchy (pg. 85 col. 2 1st full par. "exploits the hardware performance counters").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to implement Luk's profiling (pg. 43, the par. bridging the cols. "based on profiling information") using Ammons methods (pg. 86, col. 1, last full par.; pg. 85 col. 2 1st full par.) Those of ordinary skill in the art would have been motivated to do so in order to achieve the improved profiling disclosed (Ammons pg. 85, col. 2, 1st full par. "extends profiling techniques in two new directions").

Regarding Claim 4: The rejection of claim 3 is incorporated; further Luk discloses analyzing the one or more profiles comprises:

correlating the one or more profiles with respective source code based on the debug information (pg. 41, col. 1, the last partial par. "decide where to launch pre-execution in the program, based on ... cache miss profiling").

Luk does not disclose identifying top loads that contribute to cache misses.

Ammons teaches identifying top loads that contribute cache misses above a predetermined level as the delinquent loads (pg. 86, col. 2, 1st partial par. "1% of the paths ... account for 42 and 56% of the misses.").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to identify the top loads discussed in Ammons (pg. 86, col.2, 1st partial par.) in Luk's profile data (pg. 43, the par. bridging the cols. "based on profiling information"). Those of ordinary skill in the art would have been motivated to do so in order to balance the number of helper threads that are created with the effectiveness of each thread.

Regarding Claims 10-11: Claims 10-11 recite a computer readable medium for instructing a computer to perform the methods of claims 3-4 and are addressed similarly.

Claims 5-7, 2-14, 17-19, 22-25 and 28-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Tolerating Memory Latency through Software-

Controlled Pre-Execution in Simultaneous Multithreading Processors" by Luk (Luk) in view of "Data Prefetching by Dependence Graph Precomputation" by Annavaram et al. (Annavaram).

Regarding Claim 5: The rejection of claim 1 is incorporated; further Luk does not disclose building a dependent graph and performing slicing based on the dependent graph. Luk does disclose "a collection of schemes … have been proposed to construct and pre-execute slices" (pg. 50, col. 1, the last partial par.)

Annavaram teaches building a dependent graph that captures data and control dependencies of the main thread (pg. 1 Abstract "efficiently generates the required dependence graphs"); and

performing a slicing operation on the main thread based on the dependent graph to generate code slices, each code slice corresponding to one of the one or more delinquent loads (pg. 1 Abstract "generate the data addresses of the marked load/store instructions"; pg. 9, par. bridging col. 1 & 2 "slices are preexecuted … for early generation of load addresses").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to integrate Anavaram's dependent graph and associated slicing operation with Luk's system. Those of ordinary skill in the art would have been motivated to do so because Luk discloses “[His] approach and [Anavaram’s] can be complementary” (see Luk pg. 50, col. 2, 1st partial par.)

Regarding Claim 6: The rejection of claim 5 is incorporated; further Luk discloses selecting the code region further comprises:

computing liveness information providing communication cost between the main thread on the one of the helper threads (pg. 49, col. 2, 1st full par. “decide which address being accessed in pre-execution (and their surrounding control flow) are mostly communicated through registers.");

determining a communication scheme communicating live-in values between the main thread and the helper thread (pg. 44, Fig. 4 “Proposed instruction set extensions to support pre-execution”) according to the liveness information, wherein the live-in values are accessed in the helper thread without re-computation (pg. 49, col. 2, 1st full par. “decide which address being accessed in pre-execution (and their surrounding control flow) are mostly communicated through registers.").

Annavaram teaches limiting traversal of the dependency graph to be within the code region for the slicing operations (pg. 4, 2nd full par. “follows only the predicted control flow path from the branch predictor”);

merging two or more of the code slices into a helper thread of the one or more helper threads to minimize code duplication (pg. 9, col. 2, 1st partial par. “a chaining trigger mechanism whereby speculative threads are also allowed to spawn other speculative threads");

determining a change in size of the helper thread according to one of the slicing operations corresponding to a separate code region of the one or more overlapping code regions, wherein the code region encompasses the separate code region, wherein the change reduces the size of the helper thread (pg. 9, col. 1, 2nd full par. "uses hardware to generate its dependence graphs dynamically, which significantly reduces the graph size and makes precomputation quite feasible").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the references as discussed in the parent claim.

Regarding Claim 7: The rejection of claim 6 is incorporated; further Luk discloses selecting the code region further comprises determining a synchronization period for the helper thread to synchronize the main thread and the helper thread, the helper thread performing its tasks within the synchronization period (pg. 46, col. 1, 2nd par. a pre-execution thread must be terminated if its next PC is out of the acceptable range").

Regarding Claims 12-14: Claims 12-14 recite a computer readable medium for instructing a computer to perform the methods of claims 5-7 and are addressed similarly.

Regarding Claim 17: Luk discloses a method, comprising:

executing a main thread of an application in a multi-threading system (pg. 40, col. 1, 2nd par. “single threads running on multithreaded processor”); and

spawning one or more helper threads from the main thread having source codes including one or more code regions sharing at least one instruction in the source codes (e.g. Fig. 1(d))to perform one or more computations for the main thread when the main thread enters a code region selected from the one or more code regions having one or more delinquent loads (pg. 40, col. 1, 2nd par. “spawning helper threads … generates data addresses, on behalf of the main thread”), during a compilation of the main thread (pg. 44, col. 2, 3rd full par. “the compiler … is responsible for inserting pre-execution … performs all necessary code transformations”).

Luk does not disclose code of the one or more helper threads begin created separately from the source codes of the main thread.

Annavaram teaches one or more helper threads being created separately from source codes of the main thread (Abstract “generates the required dependence graphs at runtime … executes these graphs to generate the data addresses").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to integrate Anavaram’s dependent graph and associated slicing operation with Luk’s system. Those of ordinary skill in the art would have been motivated to do so because Luk discloses "researchers have investigated ways to pre-execute only a subset of instructions ... our approach and [Anavaram's] can be complementary" (see Luk pg. 50, bridging the cols.)

Regarding Claim 18: The rejection of claim 17 is incorporated; further Luk discloses:

creating a thread pool to maintain a list of thread contexts (pg. 42, col. 1, 1st full par. "N hardware contexts supported by the machine"); and

allocating one or more thread contexts from the thread pool to generate the one or more helper threads (pg. 41, col. 1, the last partial par. "Each thread-spawning instruction requests for an idle hardware context to pre-execute the code sequence").

Regarding Claim 19: The rejection of claim 18 is incorporated; further Luk discloses:

terminating the one or more helper threads when the main thread exits the code region (pg. 46, col. 1, 3rd par. "terminate a pre-execution thread if ... the main thread has executed N instructions after passing P"); and

releasing the thread contexts associated with the one or more helper threads back to the thread pool (pg. 41, col. 2, the 1st partial par. "T will free its hardware context").

Regarding Claim 22: The rejection of claim 17 is incorporated; further Luk discloses discarding results generated by the one or more helper threads when the main thread exits the code region, the results not being reused by another code region of the main thread (pg. 41, col. 2, the 1st partial par. “results held in T’s registers are simply discarded”).

Regarding Claims 23-25 and 28: Claims 23-25 and 28 recite a computer readable storage medium for instructing a computer to perform the methods of claims 17-19 and 22 and are addressed similarly.

Regarding Claim 29: Claim 29 recites a system for performing the method of claim 17 and is addressed similarly.

Claims 20-21 and 26-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over “Tolerating Memory Latency through Software-Controlled Pre-Execution in Simultaneous Multithreading Processors” by Luk (Luk) in view of “Data Prefetching by Dependence Graph Precomputation” by Annavaram et al. (Annavaram) in view of US 7,243,267 to Klemm et al. (Klemm).

Regarding Claim 20: The rejection of claim 17 is incorporated; further Luk discloses wherein the one or more help threads are placed in a run queue prior to execution (pg. 46, col. 1, last full par. “instruction queue”), further comprising:

determining a period for each of the helper threads in the run queue, each of the helper threads being terminated from the run queue when the respective period expires (pg. 46, col. 1, 2nd par. "Once this limit is reached, the thread will be terminated anyway").

The Luk-Annavaram combination does not explicitly disclose the period is a time period.

Klemm teaches determining a time period for a thread (col. 5, lines 57-58 "thread execution time exceeds user-specified threshold").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to terminate one of Luk's helper threads after a time period expires (Klemm col. 5, lines 57-58 "thread execution time exceeds user-specified threshold") as an alternate or additional instance of Luk's "system-enforced terminating conditions for preserving correctness or avoiding wasteful computation" (col. 46, col. 1, 1st par.)

Regarding Claim 21: The rejection of claim 20 is incorporated; further Luk discloses each of the helper threads terminates when the period expires even if the respective helper thread has not been accessed by the main thread (pg. 46, col. 1, 2nd par. "the thread will be terminated anyway").

Regarding Claims 26-27: Claims 26-27 recite a computer readable medium for instructing a computer to perform the methods of claims 20-21 and are addressed similarly.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Mitchell whose telephone number is (571)272-3728. The examiner can normally be reached on Monday-Thursday and alternate Fridays 7:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Bullock Lewis can be reached on (571) 272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Jason Mitchell/
Examiner, Art Unit 2193